The ACTS Collection
Robust and High-Performance Tools for Scientific Computing
Guidelines for Tool Inclusion and Retirement

Tony Drummond and Osni Marques
Lawrence Berkeley National Laboratory
One Cyclotron Road, MS 50F-1650, Berkeley, CA 94720-8139
acts-support@nersc.gov

It takes about ten years for the power of the supercomputers of any give era to migrate to the desktop. By the time we celebrate the twentieth anniversary of Computers in Physics, we can expect machines with hundreds of processors to be routinely available for scientists' personal use. Making effective use of all this horsepower will require contributions from many different fields: from the computer science and numerical analysis communities, libraries of software to distribute problems over multiple processors; from computer vendors, optimizing compilers for a multiprocessor environment; and from users, a willingness to change our way of doing business… We need to move away from a coding style suited for serial machines, where every microstep of an algorithm needs to be though about and explicitly coded, to a higher-level style, where the compiler and library tools take care of the details. And the remarkable thing is, if we adopt this higher-level approach right now, even on today's machines, we will see immediate benefits in our productivity.

W. H. Press and S. A. Teukolsky, 1997, in "Numerical Recipes: Does this Paradigm have a Future?"

1. Introduction

During the past decades there has been a continuous growth in the number of physical and societal problems that have been successfully studied and solved by means of computational modeling and simulation. Distinctively, a number of these are important scientific problems ranging in scale from the atomic to the cosmic. For example, ionization is a phenomenon as ubiquitous in modern society as the glow of fluorescent lights and the etching on silicon computer chips; but it was not until 1999 that researchers finally achieved a complete numerical solution to the simplest example of ionization, the collision of a hydrogen atom with an electron. On the opposite scale, cosmologists have long wondered whether the expansion of the Universe, which began with the Big Bang, would ever reverse itself, ending the Universe in a Big Crunch. In 2000, analysis of new measurements of the cosmic microwave background radiation showed that the geometry of the Universe is flat, and thus the Universe will continue expanding forever.

Both of these discoveries depended on high performance computer simulations that utilized computational tools included in the Advanced Computational Software (ACTS) Collection. The ACTS Collection Project evolved from the ACTS Toolkit Project, which was an umbrella project that brought together a number of general-purpose computational tool development projects funded and supported by the U.S. Department of Energy (DOE). These tools, which have been developed independently, mainly at DOE laboratories, make it easier for scientific code developers to write high performance applications for parallel computers. They tackle a number of computational issues that are common to a large number of scientific applications,

Category	*Tool*	*Functionalities*
Numerical	**Aztec**	Algorithms for the iterative solution of large sparse linear systems.
	Hypre	Algorithms for the iterative solution of large sparse linear systems, intuitive grid-centric interfaces, and dynamic configuration of parameters.
	PETSc	Tools for the solution of PDEs that require solving large-scale, sparse linear and nonlinear systems of equations.
	OPT++	Object-oriented nonlinear optimization package for serial architectures.
	PVODE	Solvers for the solution of systems of ordinary differential equations, nonlinear algebraic equations, and differential-algebraic equations.
	ScaLAPACK	Library of high performance dense linear algebra routines for distributed-memory message-passing.
	SuperLU	General-purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations.
	TAO	Large-scale optimization software, including nonlinear least squares, unconstrained minimization, bound constrained optimization, and general nonlinear optimization.
Code Development	**Global Arrays**	Library for writing parallel programs that use large arrays distributed across processing nodes and that offers a shared-memory view of distributed arrays.
	Overture	Object-Oriented tools for solving computational fluid dynamics and combustion problems in complex moving geometries.
Code Execution	**CUMULVS**	Framework that enables programmers to incorporate fault-tolerance, interactive visualization and computational steering into existing parallel programs.
	Globus	Bag of services for the creation of computational Grids and tools with which applications can be developed to access the Grid.
	PAWS	Framework for coupling parallel applications within a component-like model.
	SILOON	Tools and run-time support for building easy-to-use external interfaces to existing numerical codes.
	TAU	Set of tools for analyzing the performance of C, C++, Fortran and Java programs.
Library Development	**ATLAS and PHiPAC**	Tools for the automatic generation of optimized numerical software for modern computer architectures and compilers.
	PADRE	C++ layer for interfacing with libraries that distribute data on parallel computers.

Table 1. Current ACTS Collection Tools and a brief description their functionalities. For more details visit the ACTS information center at *http://acts.nersc.gov/tools*.

mainly implementation of numerical algorithms, and support for code development, execution and optimization. The ACTS Collection Project enables the use of these tools by a much wider community of computational scientists, and promotes code portability, reusability, reduction of duplicate efforts, and tool maturity.

By analogy with a commercial company, the ACTS Project has put the formerly uncoordinated research and development performed at various places into one distribution channel. ACTS complements the R&D effort by adding technical support, quality assurance, and marketing. Specifically, ACTS has provided technical support, tool information and evaluations. Consequently, it has increased the availability and stability of the tools; and has educated scientists about the Collection through presentations and workshops. These efforts have resulted in increased use of the tools in significant scientific applications. Feedback from users has resulted in improvements in the tools.

Currently, the ACTS Collection comprises 17 tools (see Table 1) that provide implementations of numerical algorithms, support for code development, run time environments, and library development. This paper presents a list of guidelines for the inclusion and retirement of tools into the ACTS Collection. The primary goal is to provide a robust set of high performance tools that best meets the software requirement of computational scientists and their applications.

In this document, we propose a mechanism for the inclusion of new tools into the Collection with a yearly peer-reviewed process that certifies the tools as part of the ACTS Collection. At the end of a fiscal year, tools should present progress reports (new version, target applications, etc.) and a certification panel would judge which tools to include, which to retired and which need further development.

2. Why should there be an ACTS Collection?

Our initial work with the ACTS Toolkit (an umbrella project sponsored by the MICS office under the DOE 2000 initiative) included many successful interactions with the scientific computing community, including tool users and developers. From the outcome of these interactions we have learned various lessons that were taken in consideration during the design of the ACTS Collection Project. Further these lessons must also be taken into consideration during the implementation and evolution of the ACTS Project. Here we summarize the lessons that are relevant to the continuation of the ACTS Project and the certification of tools:

- *There is still a gap between tool developers and application developers which leads to duplication of efforts*. Without projects like ACTS, application developers will continue to design and implement codes using techniques that are already available from other sources. Quite often these new developments are far from optimal because of the application developers' inexperience with all the different issues that lead to optimal performance. In many cases, application developers only consult sources like Numerical Recipes [1] that do not address platform optimization and parallelism, algorithm robustness, and language specific optimization issues. The ACTS Collection Project has been carefully designed to vitalize the current development efforts to promote reusability of quality and reliable software.

- *Users demand long-term support of the tools*. One of the main concerns that users have expressed to us is the longevity of support from tool developers and required evolution of the software as the hardware technology continues to evolve and the complexity of the scientific application continues to grow. Inasmuch as a solid base collection of reusable tools is utilized inside newer tool developments, users are guaranteed the evolution of the tools; thus long-term support and functionality are also guaranteed

- *Applications and users play an important role in hardening tools*. The main parameters for maturity are portability, robustness, acceptance, and long-term support. It is particularly the interactions with real users and real applications that have made the software mature, portable, robust and better documented. In turn, mature software will be widely accepted inside a given scientific community. The current ACTS Collection has promoted the tools to a wider national and international audience, thus increasing not only the visibility of the tools worldwide but also the range of users and applications.

- *Tools evolve or are superseded by other tools*. As technology continues to advance, there are some tool functionalities that are either no longer needed or are improved as direct consequences of the user demands. An example of these changes in the ACTS Collection is the Aztec library being superseded by AztecOO, which is one of the components of the Trilinos solver framework [2]. A project like the ACTS Collection provides mechanisms to help users with the transition and adoption of the new tools.

- *There is a demand for tool interoperability and more uniformity in the documentation and user interfaces*. Users want to experiment with functionalities available in a subset of the ACTS tools, and finding similar user interfaces and comparable levels of support and documentation makes this task even simpler and risk free. Furthermore, the computational challenges at hand demand new software developments that interact with legacy code practices, data and computer languages. ACTS provides a natural infrastructure to put in practice all the code, data and language interoperability required by these new challenges.

3. Tool Certification

In 1999, the PITAC Report [3] recommended the creation of a national library of certified domain-specific software in order to reduce the labor required for software development, testing and evolution (some of those recommendations were later implemented by the Networking and Information Technology Research and Development Program [4]. From interactions with many computational scientists and industry we have realized that ACTS has the potential of becoming a major player in such an activity. Computer vendors and a great part of the scientific computing community are already familiar with at least a couple of the ACTS tools. The implementation of the peer-reviewed certification will push the frontiers of the tools forward, and the tools or subsets of the tools will be eventually integrated in vendor-supported software libraries (for example, ScaLAPACK, one of the numerical tools in the ACTS Collection, has already been incorporated in IBM's and Cray's scientific libraries).

We seek to work with tool developers and leading computational scientists in defining the criteria to be used in the certification process. In this paper we present an initial set of parameters to carry out such a process. These parameters come from compilations of user feedback and our interactions with ACTS tools developers and their users. Further, we propose a yearly process for reviewing the status of the ACTS Collection. Tool development projects will be encouraged to participate. The matured and certified tools in the ACTS Collection will continue to set higher standards for high performance software development and continue to build the scientific community's confidence in the tools. The accumulated expertise, that will be made available through the ACTS information center, will also provide assistance to other tool developers to cope with these advancements. This approach also encourages a natural evolution for the basic set of tools to adapt to arising scientific challenges and new technology, thus promoting long-term and continuous support to users in the computational sciences community.

The ACTS Collection software certification process should be defined in terms of software correctness, robustness, functionality, portability, documentation, availability, and interoperability. The goal is not to preclude new software paradigms but to promote existing and new paradigms based on what works best for the computational science community. In the following paragraphs, we briefly define the basic set of parameters.

3.1 Correctness

This is defined by the ability of a software tool to accurately implement an algorithm or functionality. When possible, the correctness of a tool will be compared against other tools with similar functionalities. In addition tool developers must provide proofs of correctness.

We perform independent software evaluations of the tools in the ACTS Collection. These tests include the verification of correctness of their functionality. In addition, we collect feedback from users on the tools that are shared with the individual tool developers and the entire scientific computing community through the ACTS Information Center.

3.2 Robustness

Refers to the ability of a software tool to function correctly even in abnormal conditions (for example, the use of handlers and error signaling in the event of hardware violation or numerical exceptions). The robustness of a tool must not depend on a particular computational environment. By computational environment we define a combination of computer architectures and supporting software environments.

The current set of tools in the ACTS Collection reflects these high robust standards. The tools have been rigorously tested in several computational environments. The tools implementing numerical algorithms have been successfully verified and articles have been published in reputable journals in the scientific computing community. Likewise, the tools that facilitate code development and execution have been tested in a variety of computational environments and high performance applications.

3.3 Functionality and Applicability

This item refers to the value provided by the tools in today and future's computer technologies. Tools developed for computational environments or requirements that no longer comply with the technology should be retired or not considered for inclusion into the ACTS Collection. For instance, some tool development projects were developed to facilitate the use a specific hardware or a mere software trend from the past. While some of these tools provided some added value to the computational science community, they may also preclude the evolution, and performance scalability of complex applications in the state of the art technology. Further, these types of tool developments represent a high risk for application developers that in most cases plan to use their codes beyond the scope of existing architectures, compilers and programming trends.

The ACTS Collection is probing to be effective by delivering software tools that evolve. This feature is instrumental in guaranteeing the long-term support of the tools and adaptability of large complex codes to today and tomorrow's computational solutions.

3.4 Portability

This item accounts for the easy at installation and porting of one tool from one computational environment to another. Tools that target a particular computer architecture are discouraged from their participation in the ACTS Collection. Portability should also emphasize the independence of a tool from compilers and operating systems. From our experience, tools that depend on specific compilers and/or operating systems are less attractive to application developers and stops short the life cycle of software reusability.

3.5 Documentation

A very important element in promoting the use of a tool is the documentation that describes the tool functionality, and its interfaces. In addition, the practice of providing proper documentation is the only vehicle for instructing users on the correct utilization of a tool, and its limitations. Adequate levels of documentation is also effective in reducing the amount of time a user will spend prototyping her/his codes using the tool.

Currently, the tools in the ACTS Collection have various different levels of documentation. In the ACTS Collection we are working to uniformly provide appropriate levels of documentation for all tools, as well as develop didactical mechanism to teach users through examples at different levels of complexity and tool expertise.

3.6 Availability and Distribution

Tools in the ACTS Collection are distributed as open source software with its accompanying documentation. Since tools have been developed mostly at different DOE laboratories, some of them require specific agreements to comply with the software licensing policies of the institution that holds the ownership of the software. Tools under the ACTS Collection must be available to a wide audience within the computational sciences and engineering community. Software tools with higher licensing restrictions that prevent them from distributing the source codes for the entire tool are discouraged. The ACTS Collection does not commercialize the distribution of the tools and in turn the tools cannot be commercialized by a third party. Commercial packages that make use of the tools will need to instruct their users to download the ACTS tools directly from their distribution site.

3.7 Interoperability

The current ACTS funding has encouraged software interoperability and has spawned a few collaborations between projects under ACTS. Interoperability may affect performance in some cases but it potentially reduces time to solution and, most important, it assures longevity of the software. We have been working on a more general solution that can be addressed by the Common Component Architecture (CCA [5]) specifications funded by the DOE Scientific Discovery through Advanced Computing (SciDAC) Program [6]. We foresee the use of these specifications as a requirement for tools to be certified by the ACTS Collection Project and this will facilitate the interaction between the tools and migration from one tool to the other when necessary.

Moreover, language choices made by the tool developers must not dictate the choice of language used by the application developers. Thus, we propose to deal with the language interoperability issue by exploring and exploiting interface language techniques like the ones used by the Babel project [7].

3.8 Extendibility

Tools should also provide opportunities for new functionalities and expansion to different research areas. When appropriate and based on user specific needs, we propose to employ our expertise to improve the features provided by a particular tool. We foresee this activity happening in collaboration with the tool developers or independently, when funding for the development of a certain tool has ceased. As an example, some users of ScaLAPACK have already expressed an interest in features that are not currently available in that library, such as tools for facilitating the block cyclic data distribution required by ScaLAPACK. Others have expressed interest in a generalized library for data representation and translation between the numerical software tools. A last example is a distributed and general-purpose coupling interface that preempts the communication bottlenecks caused by the current centralized practices.

We propose to actively support tool developers in the adoption of interoperable software solutions like the CCA to facilitate the expansion of tool functionalities and scope

4. Goals and Milestones

The ultimate goal of the software certification and interoperability will be the distribution of the ACTS tools on user demand. For now, this has remained a difficult task given the different compiler requirements, makefiles, distributions, and installations posed by the different tool developers. By composing all the requirements and specifications inside centralized mechanisms for software distribution inside the ACTS Information Center, we will be able to provide a better service to the scientific community and alleviate early difficulties related to tool installation and testing. To realize these tasks, we propose the creation of dynamically configurable scripts that will allow users to install subsets of the interoperable tools under different computing platforms using scripting languages.

The ACTS Software repository will be a collection of actual tar (tape archive) and compressed files that will contain the distribution files of supported versions of the tools and corresponding licensing agreements. For tools that require special licensing agreements, as it is currently the case for some tools in ACTS, the repository will contain actual links to the software download sites where potential users can sign and agree to special licensing requirements. The ACTS repository will provide the aforementioned automatic installation scripts for all the tools, whether the distribution files are in the local ACTS repository or in a remote location. For the latter, users will be instructed to complete the required downloads before running the installation scripts.

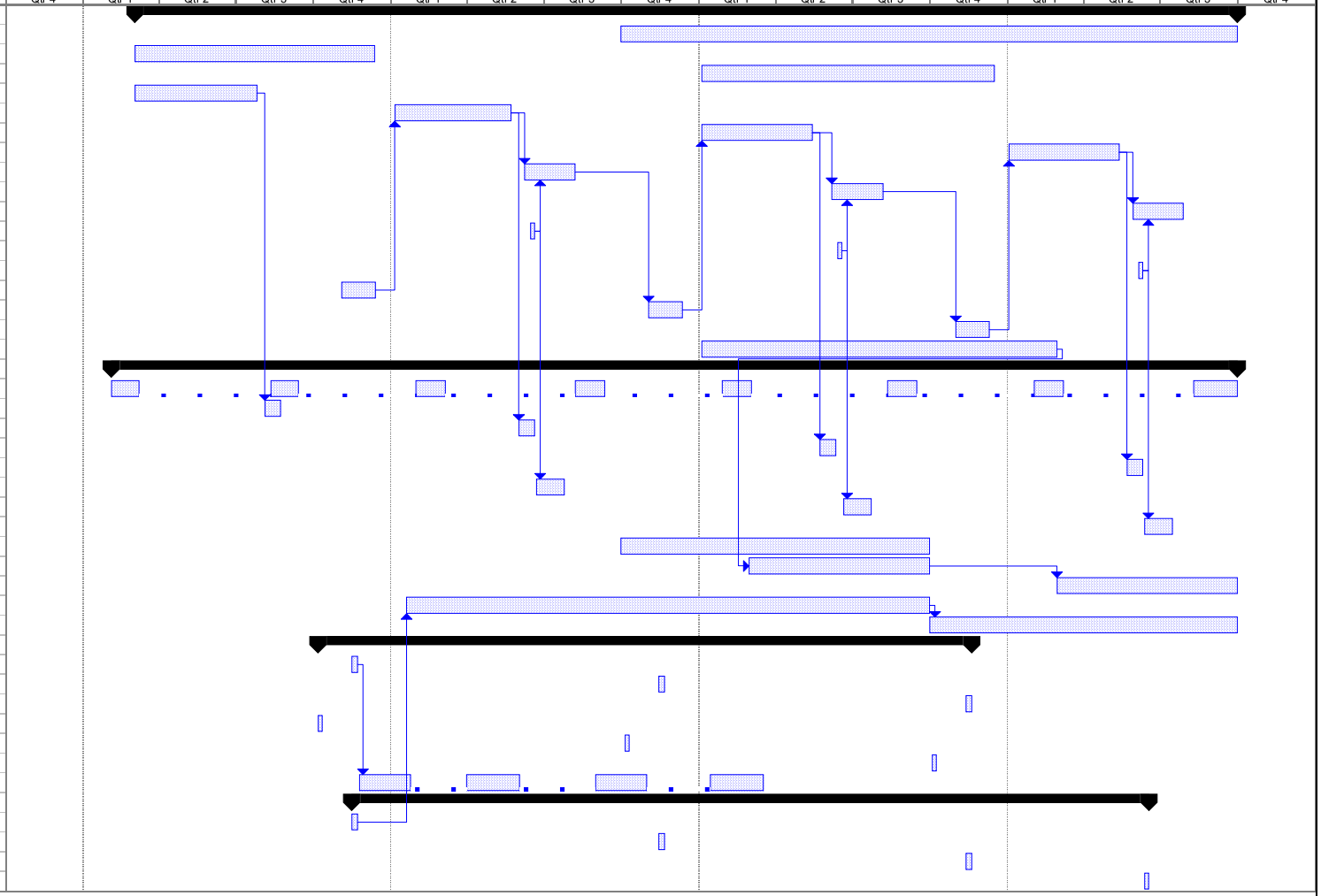
The certification process proposed here defines only a minimal set of requirements that a tool must satisfy in order to be classified as matured within the software collection. We will provide support to tool developers via the ACTS Information Center and *acts-support* to meet these requirements and shared information from the panel reviews. In addition, every time that a new version of a tool is made available, we will perform an evaluation on the applicable platforms and write the corresponding report that will be made available via the ACTS Information Center.

All these activities can be seen as part a process that has been defined as *usability engineering* [8]. The objectives of this process are basically threefold:

i. To capitalize users' knowledge so that less time is spent with software learning.
ii. To reduce possibilities of error by arranging operations in ways that match user's notions and expectations about software behavior.
iii. To facilitate user's recovery from errors resulting from improper software utilization.

We propose a list of milestones, deliverables and activities related to the issues addressed in this white paper, for the fiscal years FY03-FY05 (see attached Gantt Chart) [9]. We have strategically categorized these milestones and deliverables into four groups. The first group, *Solid Base of Tools*, contains the activities necessary to maintain and deliver a robust set of quality tools. *Information Center* refers to the activities involved in the development, management and enhancement of the ACTS Information Center. *Outreach and Education* contains a minimum set of planned activities for reaching out users of the tools. Lastly, *Collaboration with Others* refers to the interactions with other projects and institutions for the successful delivery, port and acceptance of DOE's high performance software technology by other communities.

ID	Task Name
1	**Solid Base of Tools**
2	Integration of SciDAC ISIC tools
3	Set up collaboration with ORNL/CCS
4	Set up collaborarion with ANL and PNNL
5	Tool evaluations (1st)
6	Tool evaluations (2nd)
7	Tool evaluations (3rd)
8	Tool evaluations (4th)
9	Suggestions for tool enhancements (1st)
10	Suggestions for tool enhancements (2nd)
11	Suggestions for tool enhancements (3rd)
12	Peer review panel (1st)
13	Peer review panel (2nd)
14	Peer review panel (3rd)
15	Tool certification guidelines
16	Tool certification guidelines
17	Tool certification guidelines
18	Automatic tool installation mechanisms
19	**Information Center**
20	Add/update presentations, tutorials and calendar
21	Add/update results of tool evaluations
22	Add/update results of tool evaluations
23	Add/update results of tool evaluations
24	Add/update results of tool evaluations
25	Tool certification and retirement update (1st)
26	Tool certification and retirement update (2nd)
27	Tool certification and retirement update (3rd)
28	Add bug report mechanisms to search engine
29	Automatic tool installation (1st phase)
30	Automatic tool installation (2nd phase)
31	Software and reference repository (1st phase)
32	Software and reference repository (2nd phase)
33	**Outreach and Education**
34	SC 2002 (LBNL booth)
35	SC 2003
36	SC 2004
37	Progress Report to MICS Office
38	Progress Report to MICS Office
39	Progress Report to MICS Office
40	Refereed publication (special issue of TOMS)
41	**Collaborations**
42	SC 2002 (BOF on tool certification)
43	SC 2003 (BOF on tool certification)
44	SC 2004 (BOF on tool certification)
45	Forum on future directions



Project: ACTS-Gantt-02-05
Date: Wed 11/13/02

Task
Progress
Milestone
Summary
Rolled Up Task
Rolled Up Milestone
Rolled Up Progress
Split
External Tasks
Project Summary
Group By Summary
Deadline

5. References

[1] W. Press, S. Teukolsky, W. Vetterling and B. Flannery, 1992. *Numerical Recipes in Fortran 77*, Cambridge University Press.

[2] Trilinos, *http://www.cs.sandia.gov/Trilinos*

[3] PITAC, *http://www.ccic.gov/ac/report*

[4] NITRD, *http://www.itrd.gov/pubs/blue02/index.html*

[5] CCA-Forum, *http://www.cca-forum.org*

[6] DOE/SciDAC, *http://www.er.doe.gov/scidac*

[7] Babel, *http://www.llnl.gov/CASC/components/babel.html*

[8] F. Berman, *Engineering NPACI Software for Usability*, EnVision, NPACI & SDSC, January-March 2002.

[9] O. Marques and T. Drummond, *An Expanded Framework for the Advanced Computational Software Collection, Strategic Proposal and Implementation Plan*, FY 2003-2005, October 1, 2002.